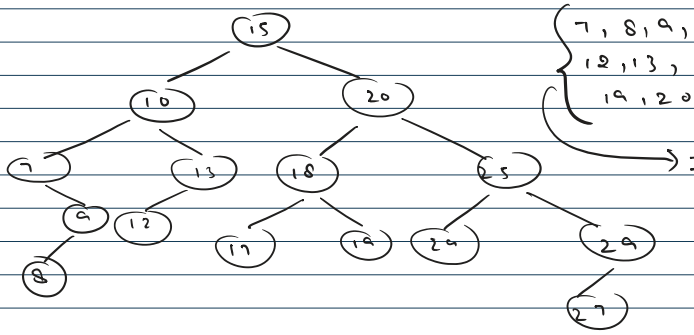
Binary Search Tree

1) BT
2) left subtree < root
3) right subtree > root

How to check if it is BST

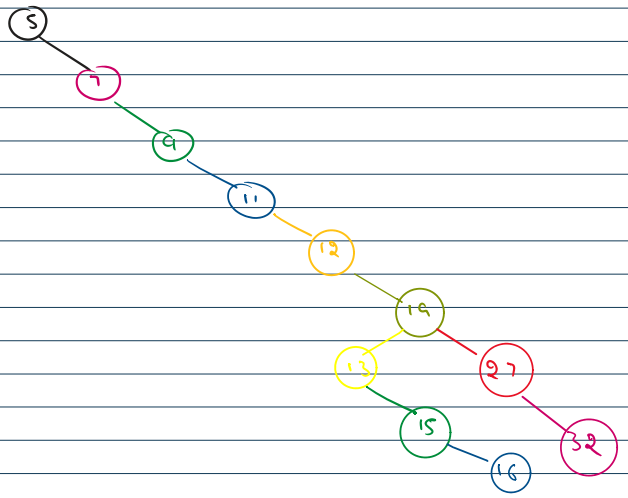


7, 8, 9, 10, 12, 13, 15, 17, 18, 19, 20, 24, 25, 27, 29

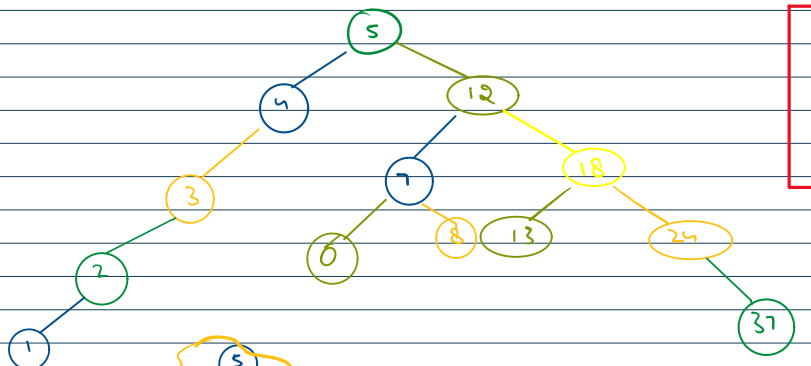
→ Inorder Traversed

Binary Search Tree construction

5, 7, 9, 11, 12, 19, 13, 27, 32, 15, 16



5, 4, 3, 12, 18, 2, 1, 24, 13, 37, 7, 8, 6



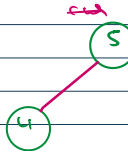
5 4
root.val > val
left side


```
private Node insertIter(Node nn, int val) {
    Node node = new Node();
    node.val = val;
    if(nn == null) {
        return node;
    }
```

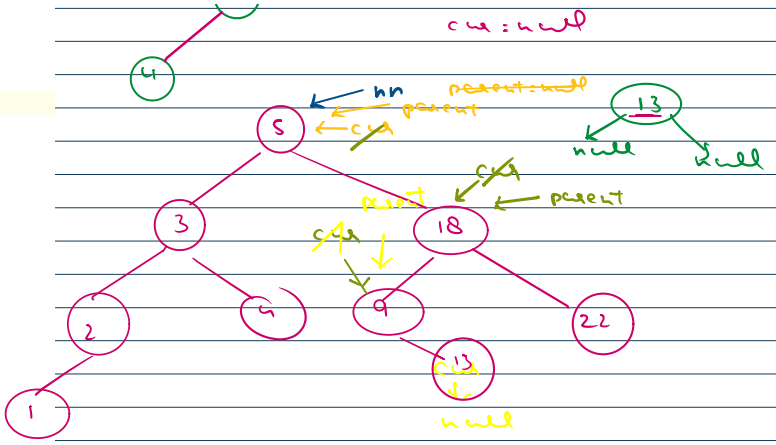
5, 4, 7, 2, 1, 8, 13, 6, 3

parent: null

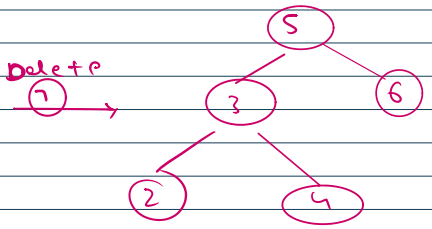
curr: null

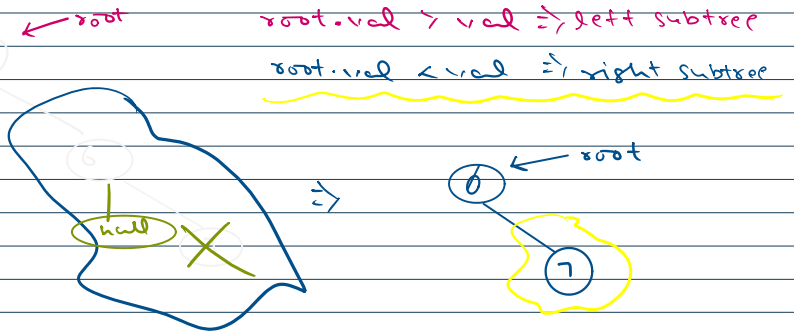


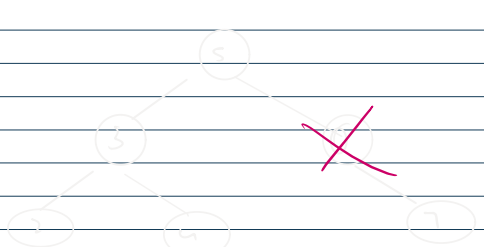
```
    node.val = val;
    if(nn == null) {
        return node;
    }
    Node cur = nn;
    Node parent = null;
    while(cur != null) {
        parent = cur;
        if(val < cur.val) {
            cur = cur.left;
        } else {
            cur = cur.right;
        }
    }
    if( val < parent.val) {
        parent.left = node;
    } else {
        parent.right = node;
    }
    return nn;
}
```

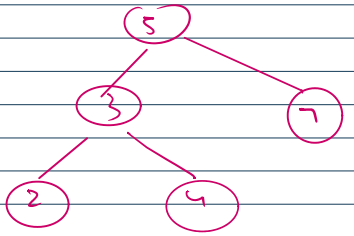


Delete Node in BST

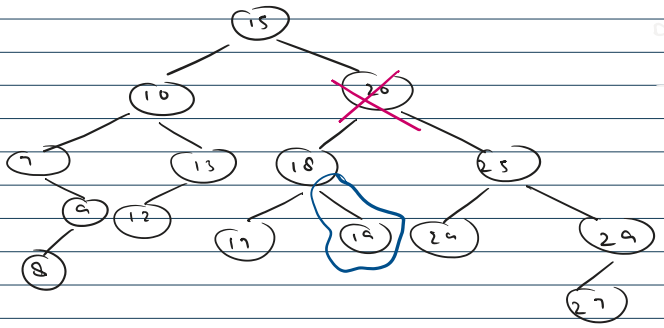


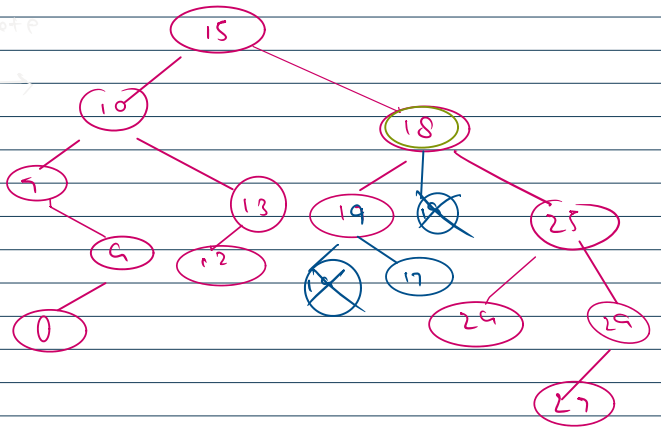


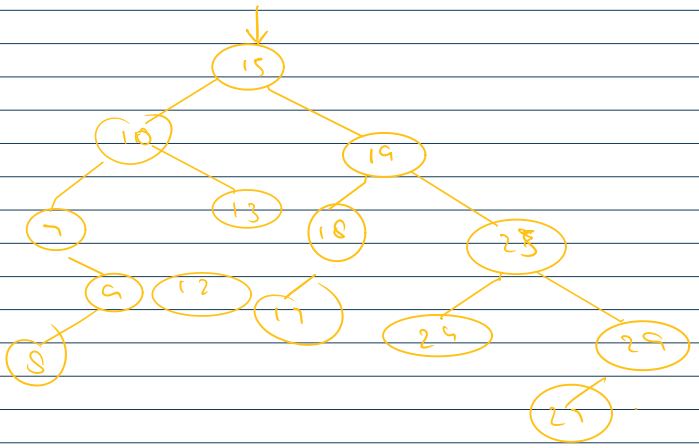


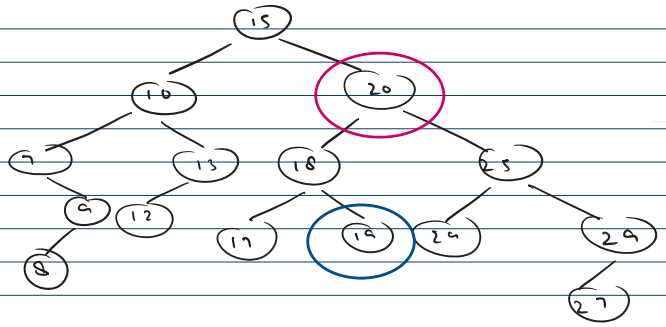


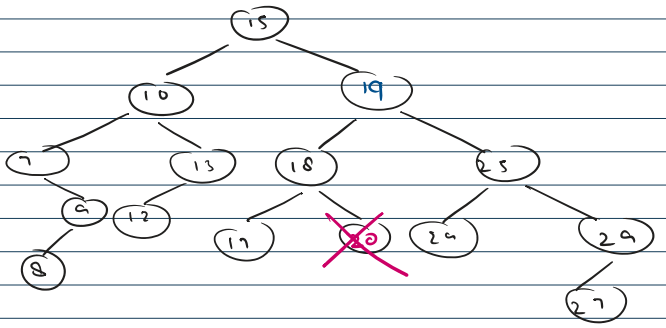

root

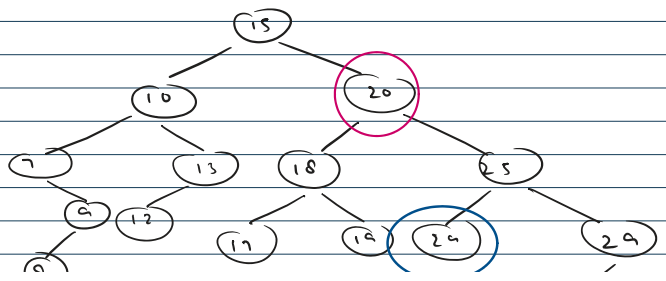

15
10
20
7
13
18
25
5
12
17
19
24
29
8
27

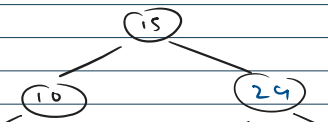

15
10
18
7
13
19
25
5
12
17
24
29
27

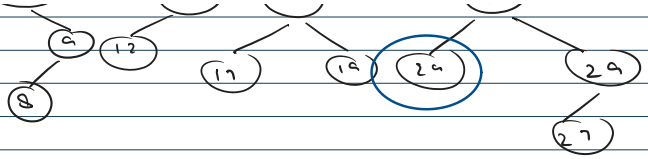

15
10
19
7
13
18
25
5
12
17
24
29
27

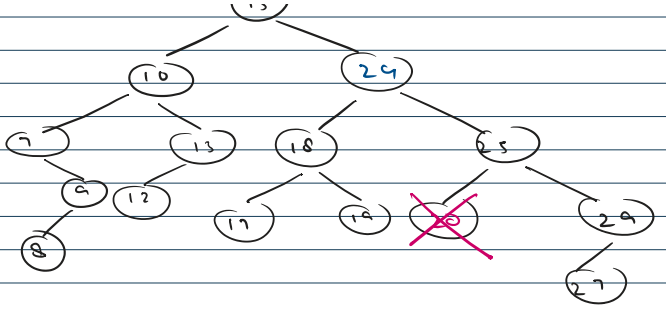

15
10
20
7
13
18
25
5
12
17
19
24
29
8
27


15
10
19
7
13
18
25
5
12
17
24
29
8
27

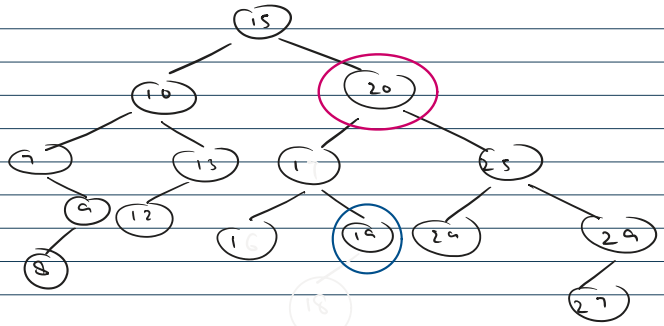

15
10
20
7
13
18
25
5
12
17
19
24
29

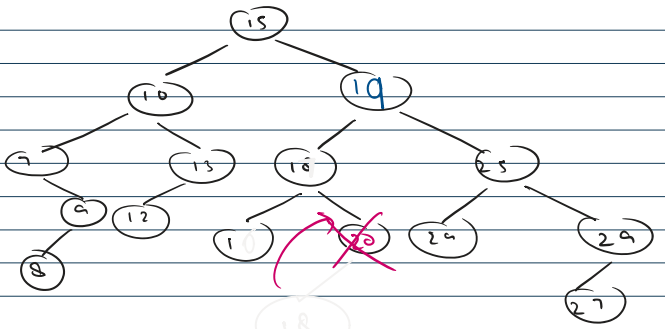

15
10
24

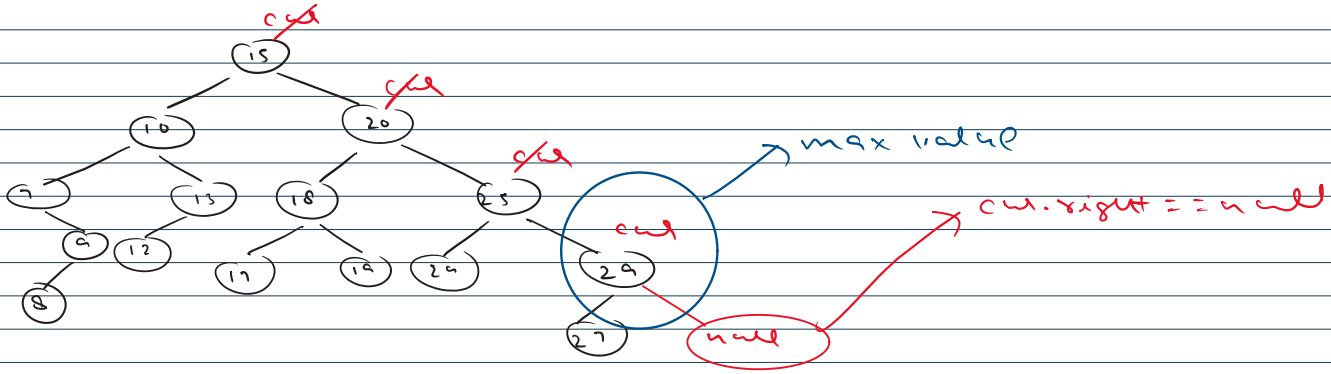


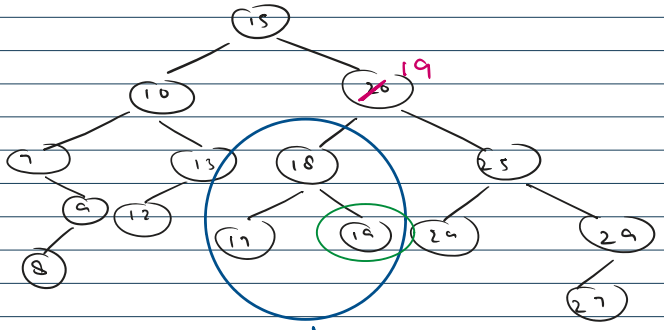


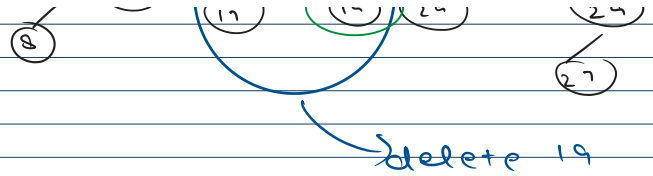
left subtree -> largest value
right subtree -> smallest value } -> swap



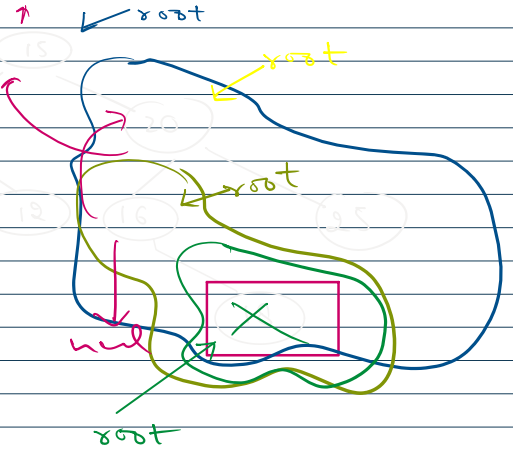








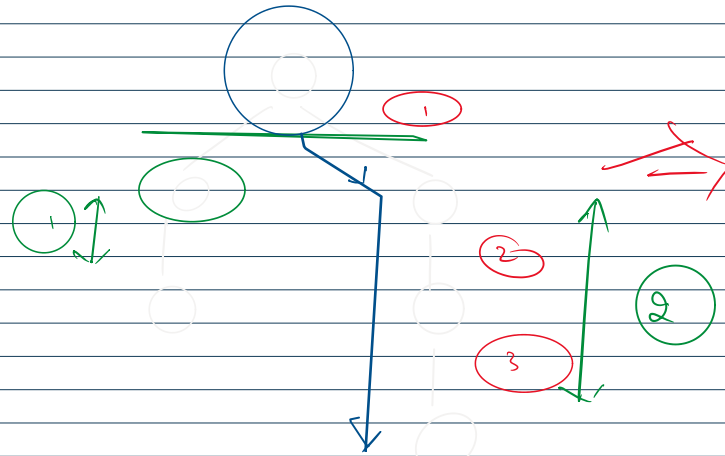

```java
public TreeNode deleteNode(TreeNode root, int key) {
    if(root == null) {
        return null;
    }

    if(root.val < key) {
        root.right = deleteNode(root.right, key);
    }
    else if (root.val > key) {
        root.left = deleteNode(root.left, key);
    }
    else {
        if(root.left == null && root.right == null) {
            return null;
        }
        else if( root.left == null) {
            return root.right;
        }
        else if(root.right == null) {
            return root.left;
        }
        else {
            int max = max(root.left);
            root.val = max;
            root.left = deleteNode(root.left, max);
        }
    }
    return root;
}
```



Diameter of a Binary Tree

⇒ 5 ⇒ diameter

⇒ 6 ⇒ diameter

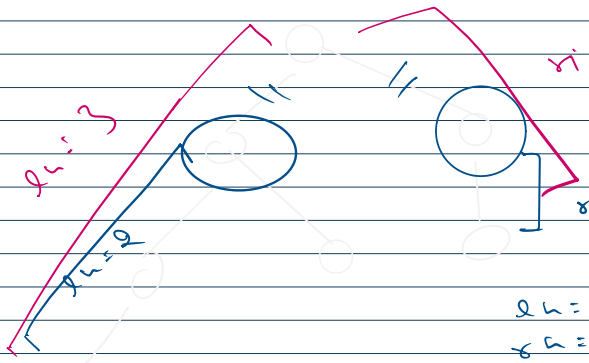


⇒ height = 3

⇒ max(2,1) + 1

⇒ height = 2

⇒ height = 0

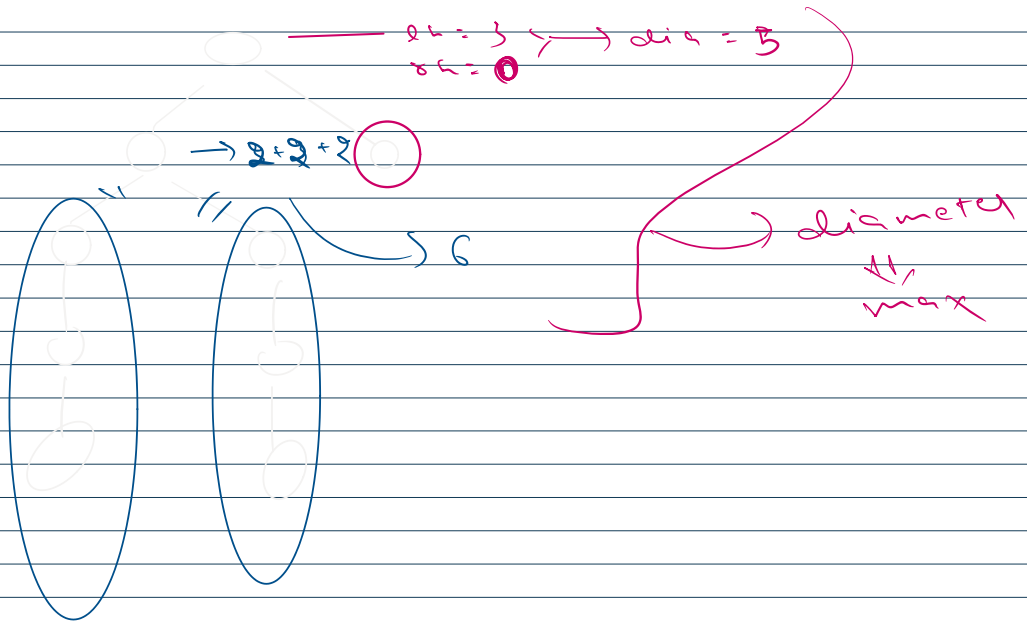


right = 2

diameter

$\{ l_h + r_h \}$

$l_h = 2$
$r_h = 1$ ⟶ diameter ⇒ $l_h + r_h + 2$


dia = 3
dia = 5
2+2+2
6
diameter
11
max